NASA Technical Memorandum 100572

THE PAWS AND STEM RELIABILITY ANALYSIS PROGRAMS

Ricky W. Butler
Philip H. Stevenson

March 1988

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

ABSTRACT

The PAWS and STEM programs are new design/validation tools. These programs provide a flexible, user-friendly, language-based interface for the input of Markov models describing the behavior of fault-tolerant computer systems. These programs produce exact solutions of the probability of system failure and provide a conservative estimate of the number of significant digits in the solution. PAWS uses a Padé approximation as a solution technique; STEM uses a Taylor series as a solution technique. Both programs have the capability to solve numerically "stiff" models. PAWS and STEM possess complementary properties with regard to their input space; and, an additional strength of these programs is that they accept input compatible with the SURE program. If used in conjunction with SURE, PAWS and STEM provide for a powerful suite of programs to analyze the reliability of fault-tolerant computer systems.

INTRODUCTION

Markov models have been used for many years to calculate the probability of failure of fault-tolerant reconfigurable computer systems. (See ref. 1.) Although fault-tree techniques are frequently employed to analyze the reliability of large complex systems, they are not powerful enough to analyze the dynamic reconfiguration capabilities of modern computer systems. However, Markov models can include both the fault recovery (e.g. via reconfiguration) and fault occurrence behaviors of such systems. The STEM and PAWS programs provide numerical solutions to user-supplied Markov models. The user defines a Markov model using a simple input language. The STEM and PAWS programs each can process the model definition and automatically calculate the state probabilities of the model for a specified time.

Reliability Modeling of Computer System Architecture

A model of a fault-tolerant system must capture the fault-occurrence and fault-recovery behavior of the system. The system is modeled with a set of states and stochastic transitions between the states. The transition times between states are governed by these stochastic distributions. If the states of the system are delineated properly, then the fault-occurrence transitions can be obtained from field data and/or by using the MIL-STD-217D Handbook calculation. These transitions have been shown to be exponentially distributed for most electronic devices (see ref. 2). The system's recovery processes can be measured experimentally using fault injection. The recovery process is typically represented as a single exponential transition. Experiments made by the Charles Stark Draper Laboratory, Inc., on the Fault-Tolerant Multiprocessor (FTMP) computer architecture have demonstrated (refs. 3, 4) that these transitions are not exponential. It is typically assumed that the error in using an exponential recovery distribution rather than the "true" distribution is negligible.[1] Sometimes several sequential states are used to model the recovery process (i.e. the method of stages). This can increase the accuracy of the method. However one must still estimate the rate parameters for each stage in the recovery process. Once a system has been mathematically modeled

---

1 In order to model the non-exponential behavior of these processes accurately, semi-Markov models are necessary.

and the state transitions determined, a computational tool such as STEM or PAWS may be used to compute the probability of entering the death states (i.e., the states that represent system failure) within a specified mission time, e.g., 10 hours.

Mathematical models of fault-tolerant systems must describe the processes that lead to system failure and the system fault-recovery capabilities. The first level of model granularity to consider is the unit of reconfiguration/redundancy in the system. In some systems this is as large as a complete processor with memory. In other systems, a smaller unit such as a CPU or memory module is appropriate. The states of the mathematical model are vectors of attributes such as the number of faulty units, the number of removed units, etc. Certain states in the system represent system failure and others represent fault-free behavior or correct operation in the presence of faults.

A semi-Markov model of a degradable quadraplex system is given in figure 1. The outputs of the processors in the quad are voted in order to mask faults.

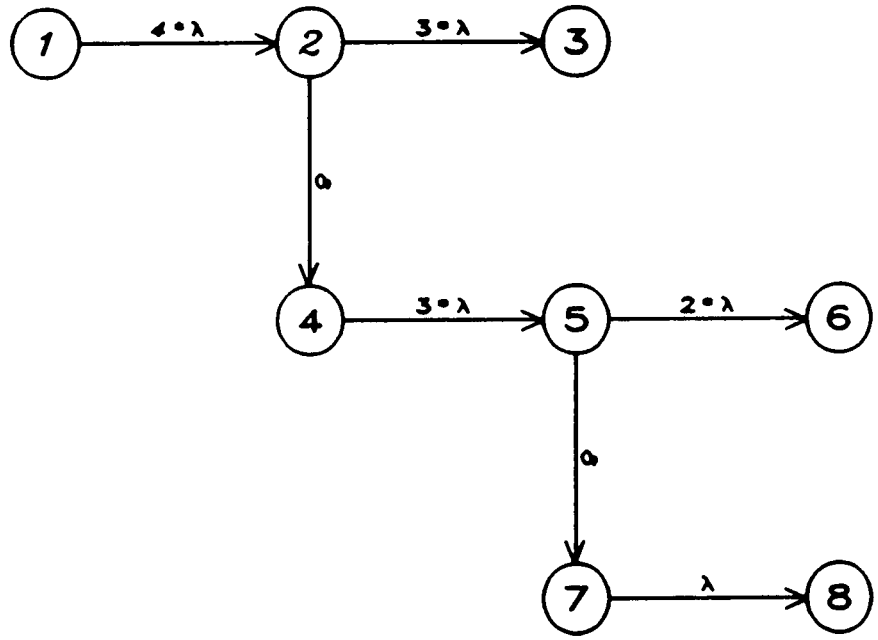

Figure 1. - Semi-Markov model of a degradable quadraplex.

The horizontal transitions represent fault arrivals. These occur with exponential rate λ. The coefficients of λ represent the number of processors

in the configuration that can fail. The vertical transitions represent recovery from a fault. The first recovery is accomplished by removing the faulty processor and degrading to a triplex. This occurs at exponential rate δ_1. The second recovery is accomplished by degrading to a simplex processor at exponential rate δ_2.

The development of a reliability model of a large, complex system uses the same concepts used in the development of the model of the degradable quad. The two types of transitions -- failure and recovery -- are still used, but there often are many different types of failure and different recoveries for each type. Thus, there may be several failure transitions from a state, each representing a failure of a different part of the system.

THE PAWS AND STEM PROGRAM

The calculation of the probability of entering a death state of a Markov model requires the solution of a set of coupled differential equations. Because of the large disparity between the rates of fault arrivals and system recoveries, models of fault-tolerant architectures inevitably lead to numerically stiff differential equations. Due to the inherent difficulties in solving such systems of equations, alternative strategies involving model decomposition-aggregation (refs. 5, 6) and path traversal probabilities (ref. 7) have been pursued. Model decomposition-aggregation methods impose limits upon the class of architectures potentially modeled. While path traversal probability methods do not suffer from this same limitation, cyclic paths can cause this method to be unsuitable. In order to resolve the problems with the alternative strategies, two numerical techniques, each specifying measures for operating with numerically stiff differential equations, have been programmed to calculate system reliability. These programs, entitled PAWS and STEM, share a common user interface which is also used by the SURE program (ref. 7, 8).

The input language to the PAWS and STEM programs is very straightforward. The input model is defined by listing all of the transitions of the model. For example, the model of figure 1 is defined as follows:

```
LAMBDA = 1E-4;          (* Failure rate of a processor *)
DELTA1 = 2.7E4;
DELTA2 = 4.7E4;
```

```
1,2 = 4*LAMBDA;
2,3 = 3*LAMBDA;
2,4 = DELTA1;
4,5 = 3*LAMBDA;
5,6 = 2*LAMBDA;
5,7 = DELTA2;
7,8 = LAMBDA;
```

The first three statements equate values to identifiers (i.e. symbolic names). The first identifier LAMBDA represents the processor failure rate. The next two identifiers DELTA1 and DELTA2 are the rates of the time to degrade to a triplex and simplex respectively. The final seven statements define the transitions of the model.

The STEM and PAWS programs currently run under VMS 4.4 on VAX-11/750 and VAX-11/780 computers at the NASA Langley Research Center. The programs have been designed with minimal usage of VMS specific constructs. Consequently, the program should be easy to transfer to other systems. Each program contains modules written in Pascal and FORTRAN. STEM's language, data, and computational modules are written in Pascal. In PAWS, the language and data modules are implemented in Pascal and the computation module is written in FORTRAN. The graphics output module for both programs is written in FORTRAN but uses the graphics library TEMPLATE; this module can be used only by installations having this library. Both PAWS and STEM can be installed and used without the graphics output module. Alternatively, this module can be rewritten using another graphics library.

The Reliability Workstation Concept

The ultra-reliability problem is too large to be completely solved by a single mathematical approach. Although different solution techniques are utilized in different programs, it is possible to have a common input language. This set of programs form the basis for a "reliability analysis workstation. The STEM and PAWS programs have been designed as part of a reliability analysis workstation based upon the SURE input language. The SURE input language is equivalent to the transition matrix of a Markov/semi-Markov model. This language is thus fully general and imposes no restriction on the type of model

that can be specified. Of course, some particular models may not be solvable by certain numerical methods. This same input language is used in the PAWS and STEM programs. The ASSIST program automatically generates Markov/semi-Markov model descriptions in this language from an abstract language(refs. 9, 10). The reliability analysis workstation structure is shown in figure 2.

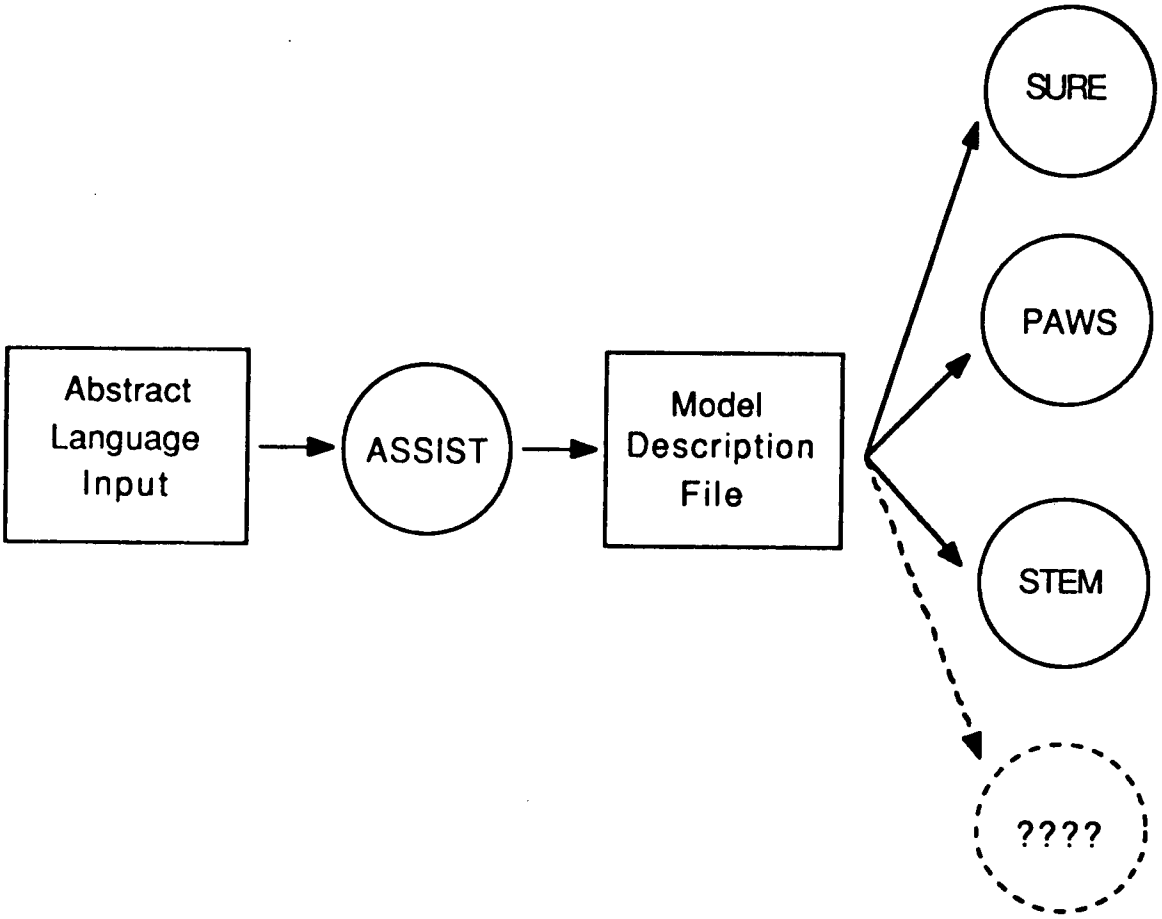

Figure 2. - Structure of Reliability Analysis Workstation.

THE FUNDAMENTAL MATHEMATICS

Mathematical Foundation of Markov Models

A discrete-space, continuous-time Markov model can be translated into an equivalent system of coupled differential equations. If the vector

$$\bar{p}(t) = \langle p_1(t),\ p_2(t),\ \ldots\ ,p_n(t)\rangle \qquad (1)$$

represents the probability of being in each of the n states of the system, and $\mathbf{A}_{n\times n} = [a_{ij}]$ is the transition matrix of the model, then the following system of differential equations represents the system:

$$\frac{d\bar{p}(t)}{dt} = \bar{p}(t)\mathbf{A} \qquad (2)$$

The solution to this system of equations is

$$\bar{p}(t) = \bar{p}(t_0)\exp(\mathbf{A}t) \qquad (3)$$

Numerical solutions to this problem have been sought and a number of methods have been proposed (ref. 11). PAWS and STEM each use a separate numerical method for determining exp(**A**t), the matrix exponential; hence, the solvers are complementary with regard to their fundamental mathematics. The numerical method for each program will be separately described. Each program will be compared in a following section to aid the user in selecting a solver appropriate to the problem under consideration.

PAWS Mathematics

The mathematical basis for evaluation of the matrix exponential in PAWS invokes the use of diagonal Pade table rational approximations in conjunction with a technique for reducing the norm of the matrix. For a detailed description of this method refer to Ward (ref. 12). Ward's description includes some extensions not utilized here.

For this method, the initial step is to reduce the $norm_1$ of matrix **A** so that it is bounded by unity, where

$$norm_1(\mathbf{X}) = \max\left(\sum_{i=1}^{n} |x_{ij}|\right). \qquad (4)$$

The technique employed results in a $norm_1$ less than or equal to one. Using an $m > 0$ such that $2^{m-1} <= norm_1(\mathbf{A}) < 2^m$, the identity,

$$\exp(\mathbf{At}) = (\exp(2^{-m}\mathbf{A}))^{2^m}, \qquad (5)$$

and

$$\mathbf{B} = 2^{-m}\mathbf{At}, \qquad (6)$$

yields:

$$\exp(\mathbf{At}) = (\exp(\mathbf{B}))^{2^m}. \qquad (7)$$

Evaluation of exp(**At**) now reduces to exponentiating a matrix(**B**) whose $norm_1$ is bounded by unity. Using the p'th diagonal Pade approximation, the matrix is exponentiated. The approximation is:

$$\exp(\mathbf{B}) = Q_p^{-1}(-\mathbf{B})Q_p(\mathbf{B}), \qquad (8)$$

where

$$Q_p(\mathbf{X}) = \sum_{k=0}^{p} c_k \mathbf{X}^k, \quad c_k = \frac{(2p-k)!p!}{(2p)!k!(p-k)!}. \qquad (9)$$

In this implementation, p equals 9. See Ward (ref. 12) for the discussion of selection of this value of p as appropriate for this purpose.

STEM Mathematics

The mathematical basis for evaluation of the matrix exponential in STEM is the truncated Taylor series. This method is attributed to Liou (ref. 13). The evaluation of the matrix exponential is:

$$\exp(\mathbf{At}) = \sum_{k=0}^{K} (\mathbf{At})^k/k! \qquad (10)$$

The series is truncated at K terms. Based upon the truncation point, the error introduced due to truncation can be determined (refs. 13, 14). The truncation error is minimized for the computer running STEM. For the current VAX implementation of double precision floating point arithmetic, the truncation error is set to 10^{-37}. For other computers it will be adjusted during porting of the program.

For the purpose of determining truncation error STEM requires $norm_1(\mathbf{At}) <= 1.0$. (See eqn. 4.) In order to scale **At** when this requirement is not initially satisfied, STEM exploits the following equality:

$$\exp(\mathbf{At}) = (\exp(2^{-m}\mathbf{At}))^{2^m} \qquad (11)$$

STEM reports an estimate of the precision of the answer. This estimate is based upon methods developed by Ward (ref. 12). This method determines error as a function of the representation of the floating point data type.

SELECTING PAWS VERSUS STEM

The choice between PAWS or STEM is dependent upon the problem a user wishes to solve. The following discussion should cover the salient points necessary to make an appropriate selection.

A primary consideration is the state size of the model. PAWS can only accept models with state size less than or equal to 300. STEM may accept models with state size less than or equal to 5000.

When considering model state size, it is important to consider its effect on execution speed. If a model satisfies the size requirements of both programs, then the general rule is that PAWS will work faster for smaller models (state size < 50), STEM works faster for larger models (state size > 100), and both execute approximately equally fast in the range of state sizes between 50 and 100. A secondary factor that also influences the relative speed of execution is the density of transitions between states in the model. For most models, this will not be an important consideration; but, if many of a model's states possess outgoing transitions that number greater than half the total state size of the model, PAWS should run faster than STEM. The difference in execution speed in these cases is a function of the algorithms and data structures utilized by each program and may vary according to the computer system on which the programs reside.

Another consideration in choosing STEM or PAWS is precision of the answer. Both programs produce precision estimates for the computed solution; however, STEM will generally calculate precision to be less than or equal to the precision estimate of PAWS. Both STEM and PAWS will terminate computation if

the answer has lost all precision, i.e. zero or less digits precision remaining. The factors which affect the precision estimate will be number of states in the model, the length of the time interval for which the solution is desired, the absolute magnitude of the transition rates, and the "stiffness" of the matrix of rate coefficients. Here, stiffness refers to the ratio of the smallest transition rate to the largest. As each of these factors increases, precision decreases. If it is found that the number of digits of precision in STEM is unacceptable, then it will be necessary to recompute the solution with PAWS.

THE PAWS/STEM USER INTERFACE

Basic Program Concept

The user of the PAWS or STEM must describe his Markov model using a simple language for enumerating all the transitions of the model. The user must first assign numbers to every state in the system. The Markov model then is described by enumerating all the transitions. This is accomplished by providing the source and destination states and the transition rate between them. The following statement provides an example:

```
1,2 = 0.0001;
```

This defines a transition from state 1 to state 2 with rate 0.0001. The program does not require any particular units, e.g., $hour^{-1}$ or sec^{-1}; however, the user must use consistent units. Although the transition-description statement described above is the key construct of the PAWS/STEM language, the flexibility of the PAWS/STEM programs have been increased by adding several features commonly seen in programming languages such as FORTRAN or Pascal.

The PAWS/STEM programs have been designed to be compatible with the input language of the SURE program (ref. 6, 7), which solves semi-Markov models. The exponential transitions are interpreted the same in PAWS/STEM as in SURE; however, the inputs defining non-exponential transitions are re-interpreted. For a summary of differences between PAWS/STEM and SURE see Appendix A.

The PAWS/STEM input language includes two types of statements -- model-definition statements and commands. These will be described in detail in the next sections.

Model-Definition Syntax

Models are defined by enumerating all of the transitions of the model.

Lexical Details. - The state numbers must be positive integers between 0 and the MAXSTATE implementation limit, usually 25,000. (This limit can be changed by redefining a constant in the PAWS/STEM programs and recompiling the source code.) The transition rates, conditional means and standard deviations, etc., are floating point numbers. The Pascal REAL syntax is used for these numbers. Thus, all the following would be accepted by the PAWS/STEM programs:

```
0.001
12.34
1.2E-4
1E-5
```

The semicolon is used for statement termination. Therefore, more than one statement may be entered on a line. Comments may be included any place that blanks are allowed. The notation "(*" indicates the beginning of a comment and "*)" indicates the termination of a comment. The following is an example of the use of a comment:

```
LAMBDA = 5.7E-4;    (* FAILURE RATE OF A PROCESSOR *)
```

If statements are entered from a terminal (instead of by the READ command described below), then the carriage return is interpreted as a semicolon. Thus, interactive statements do not have to be terminated by an explicit semicolon unless more than one statement is entered on the line.

The PAWS and STEM programs prompt the user for input by a line number followed by a question mark. For example,

```
1?
```

The number is a count of the syntactically correct lines entered into the system thus far plus the current one.

Constant definitions. - The user may equate numbers to identifiers. Thereafter, these constant identifiers may be used instead of the numbers. For example,

```
LAMBDA = 0.0052;
RECOVER = 0.005;
```

Constants may also be defined in terms of previously defined constants:

```
GAMMA = 10*LAMBDA;
```

In general, the syntax is:

```
"name" = "expression";
```

where "name" is a string of up to eight letters, digits, and underscores (_) beginning with a letter, and "expression" is an arbitrary mathematical expression as described in a subsequent section entitled "Expressions".

Variable definition. - In order to facilitate parametric analyses, a single variable may be defined. A range is given for this variable. The PAWS/STEM program will compute the system reliability as a function of this variable. If the system is run in graphics mode (to be described later), then a plot of this function will be made. The following statement defines LAMBDA as a variable with range 0.001 to 0.009:

```
LAMBDA = 0.001 TO 0.009;
```

Only one such variable may be defined. A special constant, POINTS, defines the number of points over this range to be computed. The method used to vary the variable over this range can be either geometric or arithmetic and is best explained by example. Thus, suppose POINTS = 4, then

Geometric:

```
XV = 1 TO* 1000;
```

where the values of XV used would be 1, 10, 100, and 1000.

Arithmetic:

```
XV = 1 TO+ 1000;
```

where the values of XV used would be 1, 333, 667, and 1000. **The * following the TO implies a geometric range. A TO+ or simply TO implies an arithmetic range.**

One additional option is available -- the BY option. By following the above syntax with BY "increment", the value of POINTS is automatically set such that the value is varied by adding or multiplying the specified amount. For example,

```
V = 1E-6 TO* 1E-2 BY 10;
```

sets POINTS equal to 5 and the values of V used would be 1E-6, 1E-5, 1E-4, 1E-3, and 1E-2. The statement

```
Q = 3 TO+ 5 BY 1;
```

sets POINTS equal to 3, and the values of Q used would be 3, 4, and 5.

In general, the syntax is

```
"var" = "expression" TO {"c"} "expression" { BY "increment" }
```

where "var" is a string of up to eight letters and digits beginning with a letter, "expression" is an arbitrary mathematical expression as described in the next section and the optional "c" is a + or *. The BY clause is optional; if it is used, then "increment" is any arbitrary expression.

Expressions. - When specifying transition or holding time parameters in a statement, arbitrary functions of the constants and the variable may be used. The following operators may be used:

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

The following standard functions may be used:

EXP(X)	exponential function
LN(X)	natural logarithm
SIN(X)	sine function
COS(X)	cosine function
ARCSIN(X)	arc sine function
ARCCOS(X)	arc cosine function
ARCTAN(X)	arc tangent function
SQRT(X)	square root

Both ( ) and [ ] may be used for grouping in the expressions. The following are permissible expressions:

```
2E-4
1.2*EXP(-3*ALPHA);
7*ALPHA + 12*LAMBDA;
ALPHA*(1+LAMBDA) + ALPHA**2;
2*LAMBDA + (1/ALPHA)*[LAMBDA + (1/ALPHA)];
```

Transition description. - A transition is completely specified by citing the source state, the destination state, and the transition rate. The syntax is as follows:

```
"source", "dest" = "rate";
```

where "source" is the source state, "dest" is the destination state, and "rate" is any valid expression defining the exponential rate of the transition. The following are valid statements:

```
PERM = 1E-4;
TRANSIENT = 10*PERM;
1,2 = 5*PERM;
1,9 = 5*(TRANSIENT + PERM);
2,3 = 1E-6;
```

SURE semi-Markov transition description. - The PAWS/STEM programs have been designed to interpret SURE input files. The mathematics of SURE allows the entry of fast transitions in a specialized, non-exponential form. For PAWS/STEM, the fast transitions are treated as exponential transitions. To enter a fast transition, the SURE user may use either of two methods -- White's method or Lee's method (ref. 8). For compatibility with SURE, PAWS/STEM interprets semi-Markov transitions in the form of White's method. Transitions specified using Lee's method are not interpreted and computation will not be performed by either PAWS or STEM once the RUN command is entered (See Ex. 5).

The following syntax is used for White's method:

```
"source" , "dest"  = <"mu", "sig" {, "frac" } >;
```

where

"mu" = an expression defining the conditional mean transition time, $\mu(F^*)$

"sig" = an expression defining the conditional standard deviation of the transition time, $\sigma(F^*)$

"frac" = an expression defining the transition probability, $\rho(F^*)$

and "source" and "dest" define the source and destination states, respectively. The third parameter "frac" is optional. If omitted, the transition probability is assumed to be 1.0, i.e., only one fast transition. All the following are valid (while in White's mode):

```
2,5 = <1E-5, 1E-6, 0.9>;

THETA = 1E-4;
5,7 = <THETA, THETA*THETA, 0.5>;
7,9 = <0.0001,THETA/25>;
```

A fast, non-exponential transition specified by White's method is transformed to an exponential transition by both PAWS and STEM. Thus, the following SURE input command:

```
1,2 = <MU,STD>;
```

is interpreted by PAWS/STEM as

```
1,2 = 1/MU;
```

If there is more than 1 transition from a state as shown below:

```
1,2 = <MU1,STD1,P1>;
1,3 = <MU2,STD2,P2>;          (* note: P1 + P2 = 1 *)
```

then these are interpreted as the following exponential transitions:

```
1,2 = P1/MU1;
1,3 = P2/MU2;
```

By default, the SURE program assumes that the White method will be used. If Lee's method is desired, the LEE command must be issued prior to entering any fast transition. If the LEE command is issued in PAWS/STEM, no solution will be produced and an error message will appear after the RUN command is issued. (See Appendix A.)

SURE FAST exponential transition description. - This transition description is used in the SURE program and is included here to aid compatibility of PAWS/STEM with SURE input files. This description is needed in SURE when a user wishes to specify a fast transition using an exponential rate. The syntax is

```
"source" , "dest" = FAST "rate";
```

and will result in special processing of this transition in SURE. In PAWS/STEM, this transition description is treated as any other exponential transition description.

Commands

Two types of commands have been included in the user interface. The first type of command is initiated by one of the following reserved word:

EXIT	READ	INPUT	RUN	SHOW	IF
CALC	ORPROB	DISP	SAVE	GET	PLOT

The second type of command is invoked by setting one of the following special constants

ECHO LIST POINTS START TIME

equal to one of its pre-defined values.

EXIT command. - The EXIT command causes termination of PAWS or STEM programs.

READ command. - A sequence of PAWS/STEM statements may be read from a disk file. The following interactive command reads PAWS/STEM statements from a disk file named SIFT.MOD:

```
READ SIFT.MOD
```

If no file name extent is given, the default extent .MOD is assumed. A user can build a model description file using a text editor and use this command to read it into either PAWS or STEM.

INPUT command. - This command increases the flexibility of the READ command. Within the model description file created with a text editor, INPUT commands can be inserted that will prompt for values of specified constants while the model file is being processed by the READ command. For example, the command

```
INPUT LVAL;
```

will prompt the user for a number as follows:

```
LVAL?
```

and a new constant LVAL is created that is equal to the value input by the user. Several constants can be interactively defined using one statement, for example:

```
INPUT X, Y, Z;
```

RUN command. - After a model has been fully described to PAWS/STEM, the RUN command is used to initiate the computation:

```
RUN;
```

The output is displayed on the terminal according to the LIST option specified. If the user wants the output written to a disk file instead, the following syntax is used:

```
RUN "outname";
```

where the output file "outname" may be any permissible VAX VMS file name. Two positional parameters are available on the RUN command. These parameters enable the user to change the value of the special constants POINTS and LIST in the RUN command. For example

```
RUN (30,2) OUTFILE.DAT
```

is equivalent to the following sequence of commands:

```
POINTS = 30;
LIST = 2;
RUN OUTFILE.DAT
```

Each parameter is optional so the following are acceptable:

```
RUN(10);        -- change POINTS to 10 then run.
RUN(,3);        -- change LIST to 3 and run.
RUN(20,2);      -- change POINTS to 20 and LIST to 2 then run.
```

After a run is completed, the PAWS/STEM program clears all of the transition, constant and variable definitions, returning the program state to its original state. However, throughout the session, the output of each RUN is stored internally. The results of prior "RUNS" are available in special variables

which can be referenced in future model descriptions or in a CALC command. The syntax is as follows:

#1 -- solution for RUN #1 (no variable)

#2[2] -- solution for second value of variable on RUN #2

SHOW command. - The value of a constant or variable may be displayed by the following command:

```
SHOW ALPHA;
```

Information about a transition may also be displayed by the SHOW command. For example, information concerning the transition from state 654 to state 193 is displayed by the following command:

```
SHOW 654-193;
```

More than one constant, variable, etc. may be shown at one time:

```
SHOW ALPHA, 12-13, BETA;
```

IF command. - The IF statement provides a "conditional assembly" capability to the PAWS and STEM programs. The statement following the THEN reserved word is only processed if the preceding boolean expression is true. The syntax of this statement is:

```
IF "expression" "bool-op" "expression" THEN "statement";
```

where

"bool-op" is one of the following operators: = < <= > >=

The following session illustrates this command:

```
$ PAWS

  1? X = 1; Y = 2;
  2? IF X = 1 THEN Y = 3;
     Y        CHANGED TO 3.00000E+00
  3? SHOW Y;
     Y         = 3.00000E+00
  4? IF Y > X THEN 1,2 = 1E-4;
  5? SHOW 1-2;
     TRANSITION 1 -> 2: EXPONENTIAL RATE = 1.00000E-4;
  6? IF X < 0 THEN 2,3 = 1E-3;
  7? SHOW 2-3
     TRANSITION 2 -> 3 NOT FOUND
  8? EXIT
```

CALC command. - For convenience, a calculator function has been included. This command allows the user to obtain the value of an arbitrary expression. For example, if the following commands are entered:

```
X = 1.6E-1;
CALC  (X-.12)*EXP(-0.001) + X**3;
```

the system responds with:

```
= 4.405601999335E-02
```

If a variable has been defined prior to issuing the CALC function, the expression is computed as a function of the variable over the specified range. The PLOT command can be used after the CALC command to obtain a plot of the function. The output can be sent to a disk file instead of the terminal by using the following syntax:

```
CALC "expression" TO "filename";
```

where "filename" is the name of the destination file.

ORPROB command. - A common complaint about the Markov approach to modeling is the rapid growth in state space size as the complexity of a system is increased. For large, complex inter-dependent systems, this is often unavoidable. But, systems which consist of several isolated subsystems can be analyzed easily using the additive law of probability.

Suppose the probabilities that subsystem 1 and subsystem 2 fail within the mission time are P_1 and P_2, respectively. If these subsystems fail independently, the probability of system failure, P_{sys}, can be calculated as follows:

$$P_{sys} = P_1 + P_2 - (P_1)(P_2).$$

If there are failure dependencies between the subsystems, then a single model must be used.

The ORPROB command lists all of the previous run output results and then computes the probabilistic OR of the previous runs. See example 4 of the section entitled "Examples". The PLOT command may be used to plot the results of the ORPROB command. If the variable feature of PAWS/STEM is used and LIST = 1, then the ORPROB command does not list out the answers from the previous runs. Only the probabilistic OR for each value of the variable is given. If LIST = 2 is set prior to issuing ORPROB, then a detailed list of all the outputs from the previous runs, along with the probabilistic OR of the runs for each value of the variable, is given.

ECHO constant. - The ECHO constant can be used to turn off the echo when reading a disk file. The default value of ECHO is 1, which causes the model description to be listed as it is read. (See example 4 in the section entitled "Example Sessions.")

LIST constant. - The amount of information output by the program is controlled by this command. Four list modes are available as follows:

LIST = 0; No output is sent to the terminal, but the results can still be displayed using the PLOT command.

LIST = 1; Only the probability of total system failure is listed. This is the default.

LIST = 2; The probability for each death state in the model is reported along with the total probability of entering a death state.

LIST = 3; The probability for each death state in the model is reported along with the total probability of entering a death state. The probability for all states in the model is then reported.

If a variable is defined, output according to the list mode is given for each value of the variable.

POINTS constant. - The POINTS constant specifies the number of points to be calculated over the range of the variable. The default value is 25. If no variable is defined, then this specification is ignored.

START constant. - The START constant is used to specify the start state of the model. If the START constant is not used, the program will use the source state (i.e. the state with no transitions into it) of the model (if one exists.) If there is no source state in the model, the program will use the first state entered as the start state. If no start state is specified and there are two or more source states, an error message is issued. The program arbitrarily chooses one of the source states as the start state and proceeds.

TIME constant. - The TIME constant specifies the mission time. For example, if the user sets TIME = 1.3, the program computes the probability of entering the death states of the model within time 1.3. The default value of TIME is 10. All parameter values must be in the the same units as the TIME constant.

PAWS/STEM Graphics

Although the PAWS/STEM programs are easily used without graphics output, many users desire the increased user-friendliness of the tool when assisted by graphics. The NASA Langley Research Center's AIRLAB contains four Megatek color graphics monitors (and TEMPLATE support software) enabling the full utilization of PAWS/STEM's graphics capability. However, the version of PAWS/STEM available from COSMIC does not contain the graphics software. The PAWS/STEM program can plot the probability of system failure as a function of any model parameter as well as display the semi-Markov models in a graphical

form. The output from several PAWS/STEM runs can be displayed together in the form of contour plots. Thus, the effect on system reliability of two model parameters can be illustrated on one plot. The generation of a graphical picture of the semi-Markov model can be directed by user input or left completely to the PAWS/STEM program.

After a RUN, CALC, or ORPROB command, the PLOT command can be used to plot the output on the graphics display. The syntax is

PLOT <op>, <op>, ... <op>

where <op> are plot options. Any TEMPLATE "USET" or "UPSET" parameter can be used, but the following are the most useful:

XLOG	plot x-axis using logarithmic scale
YLOG	plot y-axis using logarithmic scale
XYLOG	plot both x- and y-axes using logarithmic scales
NOLO	plot x- and y-axes with normal scaling

XLEN=5.0	set x-axis length to 5.0 in.
YLEN=8.0	set y-axis length to 8.0 in.
XMIN=2.0	set x-origin 2 in. from left side of screen
YMIN=2.0	set y-origin 2 in. above bottom of screen

The PLOTINIT and PLOT+ commands are used to display multiple runs on one plot. A single run of PAWS/STEM generates unreliability as a function of a single variable. To see the effect of a second variable (i.e. display contours of a 3-dimensional surface) the PLOT+ command is used. The PLOTINIT command should be called before performing the first PAWS/STEM run. This command defines the 2nd variable (i.e. the contour variable):

PLOTINIT BETA;

This defines BETA as the 2nd independent variable. Next, the user must set BETA to its first value. After the run is complete, the output is plotted using the PLOT+ command. The parameters of this command are identical to the PLOT command. The only difference is that the data is saved, so it can be

displayed in conjunction with subsequent run data. Next, BETA must be set to a second value, another PAWS/STEM run made, and PLOT+ must be called again. This time both outputs will be displayed together. Up to 10 such runs can be displayed together.

In order to obtain a graphical display of the semi-Markov model being processed, the user must issue the DISP command

```
DISP;
```

prior to entering any transition commands. This command causes the system to prompt for the state locations while the model is being defined. The user indicates by joystick input where each state of the model should be located. The system automatically pans as the model exceeds the current scope of the screen. Once the user indicates where each state should be placed, the program automatically draws all the transitions and labels them. The DISP command will be more fully explained in the following section. The user may store the state location information on disk using the SAVE command. For example, the current state location information is written to file SIFT.MEG by the following command:

```
SAVE SIFT
```

State location information may be retrieved from a disk file by using the GET command. If state location has been stored on disk file FTMP.MEG in a prior PAWS/STEM session, then the following command will retrieve this information:

```
GET FTMP
```

An abbreviation can be used if the location information is on a file with the same VMS file name (except the extent) as the command file that describes the model. For example, the commands GET TRIPLEX.MEG; READ TRIPLEX.MOD may be abbreviated as:

```
READ TRIPLEX*;
```

The extent names must be .MOD for the file containing the model commands and .MEG for the file containing the state locations on the graphics display in order for this abbreviation technique to work.

The SCAN and ZOOM commands may be used to peruse the model. The joystick button is used to end the ZOOM and SCAN commands. Each of the special graphics commands will be described in the subsequent sections.

DISP command. - The DISP command initializes the model display capability of the PAWS/STEM programs. After this command is issued, PAWS or STEM displays every transition it processes on the graphics device. The states of the model are represented by circles containing the number of the state. The transitions are represented by lines connecting the states. The determination of the best place to locate a state in the model (i.e., where to put the node of the graph) is a difficult problem (even for a human). A simplistic heurism is included in the PAWS/STEM programs to aid the user in positioning a state with the "wand joystick". This heurism can be utilized in two different ways - fully automatic or manual. In the fully automatic mode, the program places the state without prompting the user for joystick input. However, for complex models the picture is often quite ugly, with transition lines crossing in many places. In the manual mode the program selects a position and sets the cross-hairs at that location. If the user likes the location, he need only press the wand button. Otherwise, the position can be changed with the joystick prior to hitting the button. If fully automatic state location is desired, the user issues the following command

```
DISP*
```

If the manual mode is desired the command

```
DISP
```

is used.

The length of the transition selected by the heurism can be specified using parameters on the DISP command. By default the length in both the x and y direction is set at 2 inches. If the default value is not desired, the lengths can be changed as shown

```
DISP 2.5, 5.6
```

This sets the x-length to 2.5 inches and the y-length to 5.6 inches.

Finally the DISP command can be used to generate a hard-copy of the screen on the plotter via the following syntax

```
DISP COPY
```

GET and SAVE commands. - Once the locations of the states have been established using either the manual joystick input method or the automatic heuristic method, this information can be saved on a file using the SAVE command. The syntax is simply

```
SAVE "filename"
```

where "filename" is in VMS file syntax. In future sessions this information can be retrieved using the GET command:

```
GET "filename"
```

If no VMS file-name extent is given, the program assumes it to be .MEG by default. The format of the file is simple and can be edited using a text editor if desired. The format is 3 columns of numbers, with each row defining a particular state's location. The first column contains the state numbers, the second column contains the x-coordinates and the 3rd column contains the y-coordinates, for example:

```
 30    1.250000        118.7500
 31    4.250000        118.7500
 32    7.250000        118.7500
 20    4.250000        115.7500
 21    7.250000        115.7500
```

If a row is deleted by the editor, then if this file is used in a later session (i.e., using the GET command) only the deleted state's location will have to be entered via the joystick.

CLEAR command. - The CLEAR command erases all transitions and state locations from internal memory. However, the CLEAR* erases only the state locations specified as parameters plus all of the transitions. For example,

```
CLEAR* 3,7
```

erases all the transitions but retains all state locations except 3 and 7. The user can then re-issue the READ* (or DISP; READ) command and the program will only prompt for states 3 and 7. All of the other states will be located in the same place they were in the previous display.

ZOOM and SCAN commands. - The SCAN command causes the graphics view to pan across the model. The direction of the pan is in the direction the joystick is turned. When the final position is selected the wand button can be pressed to terminate the pan.

The ZOOM command causes the graphics display to "zoom in" or "zoom away" from the model. If the wand is pushed forward, the zoom is inward; If the wand is pulled backward the zoom is away from the model. This process is also terminated by pressing the wand button. At this time the program asks if a hard-copy on the plotter is desired.

```
HARD COPY?  (YES=1, NO=0)
```

After this the user is asked to select a new center point around which the program will re-expand the model to its normal size. This is accomplished using the joystick and wand button as in the scan mode.

SCREEN constant. The size of the display screen can be specified using the SCREEN constant. The default size is 10 inches by 10 inches. The display area is always square; however, the size of the square can be changed. For example if a 6 inch screen is desired the following command should issued prior to the DISP command

```
SCREEN = 6;
```

GREEK constant. The GREEK constant specifies whether constants with greek names such as LAMBDA, GAMMA, PHI, RHO, etc. should be displayed as greek characters on the display monitor (e.g. as λ, υ, ρ etc.). IF GREEK = 1 then this translation process is performed. If GREEK = 0 then this translation is not done. The default setting is GREEK = 1. Sometimes it is desired to display the model without the transitions labelled at all. This can be accomplished by setting GREEK = -1.

EXAMPLE SESSIONS

Outline of a Typical Session

The PAWS/STEM programs were designed for interactive use. The following method of use is recommended. (See example 2.)

1. Create a file of commands using a text editor describing the Markov model to be analyzed.

2. Start the program and use the READ command to retrieve the model information from this file.

3. Then, various commands may be used to change the values of the special constants, such as LIST, POINTS, etc., as desired. Altering the value of a constant identifier does not affect any transitions entered previously even though they were defined using a different value for the constant. The range of the variable may be changed after transitions are entered.

4. Enter the RUN command to initiate the computation.

Examples

The following examples illustrate interactive sessions. Unless explicitly noted, the example is applicable to both PAWS and STEM regardless of which program is used in the example. For clarity, all user inputs are given in lower-case letters.

Example 1. - This session illustrates the interactive use of PAWS/STEM.

```
$ paws

  PAWS   V5.0    NASA Langley Research Center

  1? lambda = 1e-5;
  2? 1,2 = 6*lambda;
  3? 2,3 = 5*lambda;
                 ^ IDENTIFIER NOT DEFINED
  3? 2,3 = 5*lambda;
  4? show 2-3;
  TRANSITION 2 -> 3:  EXPONENTIAL RATE =  5.00000E-05
  5? 2,4 = 1e4;
  6? 4,5 = 2*lambda;
  7? list = 2;
  8? time = 10;
  9? run

                                              ----- RUN #1

     D-STATE       PROBABILITY          ACCURACY
     -------   --------------------     --------
          3    2.9990701829745E-12
          5    5.9982802791680E-08
     -------   --------------------
     TOTAL     5.9985801861863E-08     7 DIGITS

 *** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN  0.300 SECS. CPU TIME UTILIZED

 10? exit
```

The warning message indicates that a syntax error was encountered by the program. If a user is reading a model and receives this message, he should check his input file to ensure that the model description is correct. In this example, the error was corrected in the next line and the model is correct. For a list of program-generated error messages see Appendix B.

Example 2. - The following session illustrates the normal usage of PAWS and STEM. Prior to using either program, a file TRIADP1.MOD was created with a text editor. This file contains a description of a triad system with one spare. The system uses 3-fold redundancy to mask single processor faults. If a spare is available, the system replaces a faulty processor with the spare. If no spare is available the system degrades to a simplex. For simplicity, the

means and standard deviations of both types of recovery are assumed to be the same - RECOVER and STDEV respectively. Remember that both PAWS and STEM will convert these transition descriptions to exponential descriptions with rate equal to 1/RECOVER.

```
$ paws

  PAWS   V5.0    NASA Langley Research Center

  1? read triadpl
  2: LAMBDA = 1E-6 TO* 1E-2;
  3: RECOVER = 2.7E-4;
  4: STDEV = 1.3E-3;
  5: 1,2 = 3*LAMBDA;
  6: 2,3 = 2*LAMBDA;
  7: 2,4 = <RECOVER,STDEV>;
  8: 4,5 = 3*LAMBDA;
  9: 5,6 = 2*LAMBDA;
 10: 5,7 = <RECOVER,STDEV>;
 11: 7,8 = LAMBDA;
 12: POINTS = 10;
 13: TIME = 6;

 14? run

   LAMBDA          PROBABILITY          ACCURACY       ----- RUN #1
 -----------   --------------------   --------
 1.00000E-06   1.0043471717057E-14    8 DIGITS
 2.78256E-06   8.2233323074980E-14    8 DIGITS
 7.74264E-06   7.3300711850539E-13    8 DIGITS
 2.15443E-05   7.7498137388032E-12    8 DIGITS
 5.99484E-05   1.0467121490686E-10    8 DIGITS
 1.66810E-04   1.7712901178553E-09    8 DIGITS
 4.64159E-04   3.4327803905355E-08    8 DIGITS
 1.29155E-03   7.0466715750014E-07    8 DIGITS
 3.59381E-03   1.4604501265457E-05    8 DIGITS
 1.00000E-02   2.9277064672170E-04    8 DIGITS

 3.090 SECS. CPU TIME UTILIZED
 15? exit
```

Note the default value of LIST was used which outputs the total probability of system failure for each value of the variable LAMBDA. Any parameter of a model can be used as a variable including TIME.

Example 3. - For this example, the probability of being in any state in the model is displayed via use of LIST = 3. The model is a simple Markov chain.

```
$ stem

  STEM V1.0    NASA Langley Research Center

  1? read example3
  2: LAMBDA = 1.0E-4;
  3: 1,2 = 4*LAMBDA;
  4: 2,3 = 3*LAMBDA;
  5: 3,4 = 2*LAMBDA;
  6: 4,5 = LAMBDA;
  7: LIST = 3;

  8? run

                                                ----- RUN #1

     D-STATE        PROBABILITY          ACCURACY
     -------    --------------------     --------
          5     9.9800216500101E-13

     -------    --------------------
     TOTAL      9.9800216500101E-13     13 DIGITS

      STATE         PROBABILITY
     -------    --------------------
          1     9.9600798934399E-01
          2     3.9860246375260E-03
          3     5.9820274715225E-06
          4     3.9900129883414E-09
          5     9.9800216500101E-13
-------------------------------------------------------------

 0.190 SECS. CPU TIME UTILIZED
  9? exit
```

Example 4. - The following session illustrates the use of the ORPROB command:

```
$ paws

  PAWS   V5.0    NASA Langley Research Center

  1? echo = 0
  2? read system1


  4? list = 1
  5? run
```

```
                    PROBABILITY          ACCURACY       ----- RUN #1
-----------    --------------------    --------
                6.3212055882856E-01    14 DIGITS

0.110 SECS. CPU TIME UTILIZED
 6? echo = 0
 7? read system2


 9? list = 1
10? run


                    PROBABILITY          ACCURACY       ----- RUN #2
-----------    --------------------    --------
                7.7686983985157E-01    14 DIGITS

0.030 SECS. CPU TIME UTILIZED
11? read system3

12: 1,2 = 4.0e-2;
13: 2,3 = 1.0e-1;

14? run


                    PROBABILITY          ACCURACY       ----- RUN #3
-----------    --------------------    --------
                1.2805288405490E-01    13 DIGITS

0.030 SECS. CPU TIME UTILIZED
15? orprob
   RUN #       LOWERBOUND     UPPERBOUND
----------    -----------    -----------
     1         6.32121E-01    6.32121E-01
     2         7.76870E-01    7.76870E-01
     3         1.28053E-01    1.28053E-01
----------    -----------    -----------
 OR PROB =     9.28426E-01    9.28426E-01

16? exit
```

Notice that the results are specified in terms of bounds even though PAWS/STEM give exact solutions. If the bounds are examined, they can be seen to be equal in each case. The output has been produced according to the SURE style. Note the effect of ECHO = 0. Also demonstrated is LIST = 1 default.

Example 5. - It has been noted in the text that PAWS/STEM is incompatible with a SURE model file using Lee mode semi-Markov transitions. This example demonstrates what will occur when a valid SURE model using the Lee mode is entered into PAWS/STEM.

```
$ stem

  STEM V1.0     NASA Langley Research Center

  1? read lee
  2: LEE;
  3: LAMBDA = 1E-4;
  4: RECOV = 1800.0;
  5: RHO =  3600.0;
  6: POINTS = 5;
  7: GAMMA = 10*LAMBDA;
  8: PHI = RECOV;
  9: T = RHO  + PHI;
 10: E = 2.718281828;
 11: QUANT2 = 1E-2;
 12: QPROB2 = 1.0 - E**(-RECOV*QUANT2);
 13: TIME = 10;
 14: 1,2= 3*LAMBDA;
 15: 2,3 = 2*LAMBDA+2*GAMMA;
 16: @2=<1/RECOV,QUANT2,QPROB2>;
***** STATE OUT OF RANGE
 17: 2,4 = <1.0>;
 18: 4,5 = LAMBDA + GAMMA;
 19: 1,6 =  3*GAMMA;
 20: QUANT6 = 1E-2;
 21: QPROB6 = 1.0 - E**(-T*QUANT6);
 22: @6=<1/T,QUANT6,QPROB6>;
***** STATE OUT OF RANGE
 23: 6,1 = <RHO/T>;
 24: 6,4 = <PHI/T>;
 25: 6,7 = 2*LAMBDA + 2*GAMMA;

 26? run

 *** START STATE ASSUMED TO BE 1


                    PROBABILITY         ACCURACY       ----- RUN #1
  ----------    -------------------     --------
*** LEE MODE ILLEGAL IN STEM ***

 0.030 SECS. CPU TIME UTILIZED
 27? exit
```

As can be seen, several errors occur. Most important, no result is reported; only the error message that Lee mode is inappropriate is displayed.

Example 6. - The mathematics of both PAWS and STEM require scaling of the model's transition rates in order to satisfy certain assumptions. For models in which the scaling is too severe, an accurate answer cannot be produced. What follows is an example using both PAWS and STEM and a very simple model with an excessively high transition rate. It is important to note that TIME = 10, the default value.

```
$ stem

  STEM V1.0    NASA Langley Research Center

  1? 1,2 = 1e30;
  2? run


                    PROBABILITY          ACCURACY       ----- RUN #1
  ----------    -------------------     --------
            ****COMPUTATION ABORTED****

 0.270 SECS. CPU TIME UTILIZED
  3? exit

$ paws

  PAWS   V5.0    NASA Langley Research Center

  1? 1,2 = 1e30;
  2? run


                    PROBABILITY          ACCURACY       ----- RUN #1
  ----------    -------------------     --------
                0.0000000000000E+00     0 DIGITS

 0.090 SECS. CPU TIME UTILIZED
  3? exit
```

Neither solver produces an answer to this problem; each reports this somewhat differently. STEM explicitly produces a message that denotes this problem. PAWS generates output that infers this difficulty. It is important to realize that it is the magnitude of the transition rate multiplied by the value of time that creates this condition. If the transition was respecified with a rate of 1.0e20 and TIME was set equal to 1.0e11, the result would be the same; however, if the rate remained the same and TIME was specified as a small interval, i.e. 1.0e-20, then a solution would be produced.

CONCLUDING REMARKS

The PAWS and STEM programs are new design/validation tools. These programs provide a flexible, user-friendly, language-based interface for the input of Markov models describing the behavior of fault-tolerant computer systems. These programs produce exact solutions of the probability of system failure and provide a conservative estimate of the number of significant digits in the solution. By using separate numerical solution techniques, PAWS and STEM possess complementary properties with regard to their input space. An additional strength of these programs is that they accept input compatible with the SURE program. If used in conjunction with SURE, PAWS and STEM provide for a powerful suite of programs to analyze the reliability of fault-tolerant computer systems.

APPENDIX A

SURE COMPATIBILITY

PAWS/STEM do not interpret certain commands and constants in the same manner as SURE. Unless specifically mentioned, assume that SURE constructs are supported by PAWS and STEM. The SURE special constants which do not initiate special processing by PAWS/STEM are:

AUTOFAST LBFACT PRUNE QTCALC TRUNC WARNDIG.

These constants are treated as any other user-defined constant by PAWS/STEM.

The SURE reserved word command which has an altered meaning in PAWS/STEM is:

LEE.

The LEE command causes a change in processing in PAWS/STEM different than that of SURE. If the LEE command is given in PAWS/STEM, the program will not compute an answer once the RUN command is issued and the following error message will be displayed,

*** LEE MODE ILLEGAL IN "prog" ***

where "prog" equals PAWS or STEM depending on which program is used (See Example 4.).

All transition statements interpreted by PAWS/STEM are interpreted as exponential transitions. The SURE White mode semi-Markov transition specified as

"source" , "dest" = < "mu", "sig" {, "frac" } >;

is interpreted during computation as

"source", "dest" = { "frac" } / "mu" ;

where

"mu" = an expression defining the conditional mean transition time,
"sig" = an expression defining the conditional standard deviation of the transition time,
"frac" = an optional expression defining the transition probability(if omitted "frac" = 1.0),

and "source" and "dest" define the source and destination states respectively.

As noted above the Lee method is not compatible with PAWS/STEM. The Lee method statement,

```
@ "source" = <"hmu", "quant', "prob">
```

will not be interpreted by PAWS/STEM and the following error message will be reported to the user:

```
***** STATE OUT OF RANGE -
```

Any exponential transition specified using the FAST exponential description of SURE is treated as any other transition specified with an exponential rate.

SURE provides access to the results of prior "RUNS" in a session through use of special symbols. Examples of the SURE syntax for these symbols follows:

```
#U2     --   upperbound of solution for RUN #2   (no variable)
#L1     --   lowerbound of solution for RUN #1   (no variable)

#L1[3] --   lowerbound of solution for third value of variable on RUN #1
#U2[1] --   upperbound of solution for first value of variable on RUN #2
```

PAWS/STEM, which compute exact solutions, will interpret these symbols by retrieving the exact solution. For example, if either #L2 or #U2 is used, PAWS/STEM retrieves the exact solution of the second RUN. Information requested from "RUNS" performed using a variable is handled similarly.

APPENDIX B

ERROR MESSAGES

The following error messages are generated by the PAWS/STEM programs. Some error messages may be generated by only one of either PAWS or STEM. When this occurs, it will be noted in the message's description; otherwise, assume the error message is common to both programs. The list of error messages in alphabetical order is:

ARGUMENT TO EXP FUNCTION MUST BE < 8.80289E+01 - The argument to the EXP function is too large.

ARGUMENT TO LN OR SQRT MUST BE > 0 - The LN and SQRT function require positive arguments.

ARGUMENT TO STANDARD FUNCTION MISSING - No argument was supplied for a standard function.

COMMA EXPECTED - Syntax error; a comma is needed.

CONSTANT EXPECTED - Syntax error; a constant is expected.

DIVISION BY ZERO NOT ALLOWED - A division by 0 was encountered when evaluating the expression.

ERROR OPENING FILE - <vms status> - The program was unable to open the indicated file.

FILE NAME TOO LONG - File names must be 80 or less characters.

FILE NAME EXPECTED - Syntax error, the file name is missing.

"id" CHANGED TO x - The value of the identifier id is being changed to x.

"id" CHANGED TO X TO Y - The range of the variable "id" is being changed.

"id" NOT FOUND - The system is unable to SHOW the identifier since it has not yet been defined.

IDENTIFIER EXPECTED - Syntax error, identifier expected here.

IDENTIFIER NOT DEFINED - The identifier entered has not yet been defined.

ILLEGAL CHARACTER - The character used is not recognized.

ILLEGAL INPUT VALUE - A non-numeric character was entered in response to the INPUT command prompt.

ILLEGAL STATEMENT - The command word is unknown by the system.

INPUT ALREADY DEFINED AS THE VARIABLE - An attempt was made to input a value for an identifier that was already defined as the variable.

INPUT LINE TOO LONG - The command line exceeds the 100 character limit.

INTEGER EXPECTED - Syntax error, an integer is expected.

MORE THAN ONE SOURCE STATE IN MODEL - The model entered by the user has more than one source state (i.e., a state with no transitions into it). If a start state has been specified by a START command, it is used. Otherwise, the program arbitrarily chooses a start state.

MUST BE IN "READ" MODE - The INPUT command can be used only in a file processed by a READ command.

NO RUNS MADE YET - The ORPROB command was called before any runs were made.

NUMBER TOO LONG - Only 15 digits/characters allowed per number.

ONLY 1 VARIABLE ALLOWED - Only one variable can be defined per model.

ONLY 100 VARIABLE RESULTS STORED - the ORPROB command can only process the first 100 values of the variable per run.

REAL EXPECTED - A floating point number is expected here.

SEMICOLON EXPECTED - Syntax error; a semicolon is needed.

START STATE ASSUMED TO BE x - There was no source state in the model and no start state was specified via a START command so the program arbitrarily selected x as the start state.

SUB-EXPRESSION TOO LARGE, i.e. > 1.70000E+38 - An overflow condition was encountered when evaluating the expression.

TRANSITION NOT FOUND - The system is unable to SHOW the transition because it has not yet been defined.

VMS FILE NOT FOUND - The file indicated on the READ command is not present on the disk. (Note: make sure your default directory is correct.)

0 STATES IN MODEL - The RUN command found no states in the model.

*** ERROR: INSTANTANEOUS TRANSITION AT STATE n. - One of the transitions from state n has been defined with a mean of zero.

*** LEE MODE ILLEGAL IN "prog" *** - "prog" equals PAWS or STEM depending on which program is used.

*** STATE x CANNOT MAP TO ITSELF - A state transition has been specified with the same state as both the from state and the destination state.

***** STATE OUT OF RANGE - State number has been improperly specified. Must be within range 0 to 10,000.

*** THE *CALC* EXPRESSION MUST BE ON 1 LINE - The mathematical expression processed by the CALC function must fit on one line. Constant sub-expressions can be defined prior to the CALC function and used to simplify the CALC expression.

*** VARIABLES INCONSISTENT BETWEEN RUNS - the ORPROB command cannot process the preceding runs since they did not use the same variable or the same values of the variable.

*** WARNING: REMAINDER OF INPUT LINE IGNORED - Any commands that followed the READ command on the same line were ignored.

*** WARNING: RUN-TIME PROCESSING ERRORS - Computation overflow occurred during execution.

*** WARNING: SYNTAX ERRORS PRESENT BEFORE RUN - Syntax errors were present during the model description process.

****COMPUTATION ABORTED**** - STEM exited computation due to inability to produce an accurate answer.

*** WARNING: VARIABLE CHANGED! - If previous transitions have been defined using a variable and the variable name is changed, inconsistencies can result in the values of the transitions.

= EXPECTED - Syntax error; the = operator is needed.

> EXPECTED - Syntax error; the closing bracket > is missing.

) EXPECTED - A right parenthesis is missing in the expression.

] EXPECTED - A right bracket is missing in the expression.

REFERENCES

1. Siewiorek, Daniel P.; and Swarz, Robert S.: The Theory and Practice of Reliable System Design, Digital Press, 1982, pp. 246-302.

2. Siewiorek, Daniel P.; and Swarz, Robert S.: The Theory and Practice of Reliable System Design, Digital Press, 1982, pp. 31-43.

3. Lala, Jaynarayan H.; and Smith, T. Basil, III: Development and Evaluation of a Fault-Tolerant Multiprocessor (FTMP) Computer. Volume III - FTMP Test and Evaluation. NASA CR-166073, 1983.

4. Finelli, George B.: Characterization of Fault Recovery through Fault Injection on FTMP. IEEE Transactions on Reliability, Vol. R-36, June, 1987, pp. 164-170.

5. Trivedi, Kishor; Dugan, Joanne Bechta; Geist, Robert; and Smotherman, Mark: Modeling Imperfect Coverage in Fault-Tolerant Systems. The Fourteenth International Conference on Fault-Tolerant Computing - FTCS 14, Digest of Papers, 84CH2050-3, IEEE, 1984, pp. 77-82.

6. Bavuso, S. J.; and Petersen, P. L.: CARE III Model Overview and User's Guide (First Revision). NASA TM-86404, 1985.

7. Butler, Ricky W.: The Semi-Markov Unreliability Range Evaluator (SURE) Program. NASA TM-86261, 1984.

8. Butler, Ricky W.; and White, Allan L.: SURE Reliability Analysis -- Program and Mathematics. NASA TP-2764, March 1988.

9. Butler, Ricky W.: An Abstract Language for Specifying Markov Reliability Models. IEEE Transactions on Reliability, Vol. R-35, December, 1986, pp. 595-601.

10. Johnson, Sally C.: ASSIST User's Manual. NASA TM-87735, August, 1986.

11. Moler, C. B.; and Van Loan, C.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, SIAM Review, Vol. 20, No. 4, Oct. 1978, pp. 801-836.

12. Ward, Robert C.: Numerical Accuracy of the Matrix Exponential with Accuracy Estimate, SIAM Journal of Numerical Analysis, Vol. 14, No. 4, Sep. 1977, pp. 600-610.

13. Liou, M. L.: A Novel Method of Evaluating Transient Response, Proceedings of the IEEE, Vol. 54, Jan. 1966, pp. 20-23.

14. Bickart, Theodore A.: Matrix Exponential: Approximation by Truncated Power Series, Proceedings of the IEEE, Vol. 56, May, 1968, pp. 872-873.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100572		

4. Title and Subtitle
The PAWS and STEM Reliability Analysis Programs

5. Report Date
March 1988

6. Performing Organization Code

7. Author(s)
Ricky W. Butler
Philip H. Stevenson

8. Performing Organization Report No.

10. Work Unit No.
505-66-21-01

9. Performing Organization Name and Address
NASA Langley Research Center
Hampton, VA 23665-5225

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, DC 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes
Ricky W. Butler, Langley Research Center, Hampton, Virginia.
Philip H. Stevenson, Planning Research Corporation, Hampton, Virginia.

16. Abstract

The PAWS and STEM programs are new design/validation tools. These programs provide a flexible, user-friendly, language-based interface for the input of Markov models describing the behavior of fault-tolerant computer systems. These programs produce exact solutions of the probability of system failure and provide a conservative estimate of the number of significant digits in the solution. PAWS uses a Padé approximation as a solution technique; STEM uses a Taylor series as a solution technique. Both programs have the capability to solve numerically "stiff" models. PAWS and STEM possess complementary properties with regard to their input space; and, an additional strength of these programs is that they accept input compatible with the SURE program. If used in conjunction with SURE, PAWS and STEM provide for a powerful suite of programs to analyze the reliability of fault-tolerant computer systems.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Markov Model Reliability Analysis Fault-Tolerance Reliability Modeling	Unclassified - Unlimited Star Category 65

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	43	A03